



John T. Anderson Engineering Note

Date: January 17, 2002
Rev Date: January 17, 2002

Project: CFT; AFE board
Doc. No: A1020117

Subject: The new and improved LVDS_MUX PLD firmware

Introduction

Power supply and system test issues that have cropped up on the platform show the need for reducing total +3.3V current consumption and the need to provide better test data generation from the AFE into the rest of the system. These two apparently unrelated issues converge in the LVDS_MUX programmable logic device. A significant redesign of the LVDS_MUX has occurred which manages to give more functions while reducing total power consumption.

This note will describe the new features while also providing some tutelage on how to design PLDs for less power consumption.

Device Requirements

The LVDS_MUX CPLD has to receive the 64 bits of discriminator information from one MCM, plus six bits of timing/status information, and multiplex this onto a 10-bit wide bus over seven cycles of the 53.104MHz clock. In addition, the device must be able to respond to commands from the AFE microcontroller to emit test data upon command or assert a firmware ID value on the data bus. The LVDS_MUX also provides some timing/synchronization help for the VSVX system.

System test issues brought about by architectural drift make the data path from AFE to DFE a difficult one to test. The Mixer Box (added after the initial system design) has no pickoff for the data it mixes. The DFE has very limited raw data buffering capability. The problem of sending known patterns falls upon the AFE. The original specification of the test pattern was that a pattern of walking ones and/or walking zeroes, plus DC all-ones or all-zeroes must be able to be selected at will. With the advent of the Mixer, it now appears necessary to add some kind of capability to send a sparse (few ones in a sea of zeroes) test pattern corresponding to an actual detector track. Initial attempts to fit this new logic have been only partially successful.

Power Issues

The AFEs are grouped into sets of eight boards supplied by a +3.3V, 40A supply (in addition to other voltages). This supply drives the 19 programmable logic devices (PLDs) on the board. Original power supply estimates gave a value of 4A per board, or 32A per backplane – a nice fit to the 40A supply. Recent measurements in place show that boards take a widely variant amount of +3.3V current ranging from 3.5A to 5A; this is apparently due to differences in PLD manufacture date. Since there are eight essentially identical LVDS_MUX parts on each AFE, a reduction in current draw of just this one design will pay significant benefits.

Relationships between PLD architecture and power usage

PLDs like those in the AFE consist of a sea of logic cells connected by a matrix. These parts are CPLDs (not FPGAs, which are quite different). In a CPLD, power is consumed by the number of interconnections utilized (nets) and the number of logic cells used (Product Terms). Each logic cell is designed as a flip-flop whose D input is connected to the input of an OR gate with some number of inputs. Each input of an OR gate is considered a Product Term. Thus, the number of macrocells used is a *rough* guide to power, but the only way to get a reasonable estimate is to look at the reports generated by the logic compilation process. Different architectures can have widely variant resource usage (and thus, power consumption).

The original design of the LVDS_MUX used 70 flip-flops hooked up as ten 7-bit shift registers. Each stage of each register used a 4:1 mux at the input of the flip-flop to handle the load, shift & test features demanded. A 4:1 mux by definition uses 4 Product Terms; thus, 70 bits of this structure will consume 280 PTs. This architecture is very nice from a designer's standpoint because it allows two freedoms:

1. The delay in every stage is exactly predictable and is the minimum delay of the device.
2. Since everything is latched in internal flip-flops, the amount of time the input data must be stable is minimized.
3. It works for a 132nsec, all crossings live architecture with no dead time.

These are powerful arguments for that architecture. However, tests of the AFEs have shown that the discriminator data coming from the MCM persists for quite a long time. This allows a design in which many fewer flip-flops are used and the data is multiplexed directly from the input pins into a much smaller number of flip-flops. To insure some control of timing, the pipeline is reduced not from 7-deep to 1-deep, but only to 2-deep. This also allows for a separation of multiplexing features. The first stage of the pipeline (only 10 flip-flops, because the device only sends out 10-bit data) is fronted by 7:1 multiplexers that capture the MCM data. A second rank of flip-flops have a 4:1 mux (one input is the output of the first rank of flops, the other three are for test data). This means only $(10*7) + (10*4) = 110$ PTs are used where before 280 PTs were needed *for the same logic*. A few more PTs (about 10) are also required to implement the mux control counters, for a ratio of about 120 / 280. In essence, the trade is halving the number of PT terms in exchange for the data taking two clock ticks to propagate.

The other reason for using this dual-stage pipeline is because it limits the maximum number of OR terms at any stage to seven. If everything is put into one big fat mux with only 10 flip-flops, the massive width of OR gates required will be bigger than can be well supported by the internal logic of the PLD; chances are it won't fit. By the way, this trick would not work as well in an Altera or Xilinx CPLD as it does in the Lattice because of the chip design. Lattice has native 7-wide OR gates without use of expanders; Altera requires expanders for any gate width greater than 4. Know thy chip structure!

Lattice gives a general "guesstimate" formula that's not too far off for estimating total current consumption. The original design estimates at 235mA (in reality, about 210) and the new design estimates at about 130ma. That's a difference of likely 80 mA in the real chips. At eight chips per board and eight boards per backplane, the power supply load has been reduced by some 5 Amps, or 12% of the total supply capability! Over 13 backplanes in the central tracker, a total power usage reduction of some 200 Watts is expected. Going back to the 5 Amp boards, one would now expect them to draw something more like 4.3 A, a much more comfortable number.

Details of Implementation and effects upon software and test stands

In gross terms the implementation is pretty much the same as before, but a few bits have been moved around to conserve PT usage. There is also a background software process (thanks to the Mixer again!) in which each AFE board has to have a 'personality' attached to it. In English, that means each board has a different set of LVDS_MUX firmwares that differ in the order in which the MCM bits are presented. A spreadsheet from the Mixer documentation is parsed by a Visual Basic program to generate the appropriate tables. These tables are then pasted into the appropriate section of the source code for the LVDS_MUX to create a specific 'personality'. This 'personality' is then compiled such that a library of LVDS_MUX codes are generated.

The structure of the rewritten section is different in this new cut of the PLD. The old cut-and-replace section looked like this:

```

=====
"
"
"   THIS SECTION IS MODIFIED ON A BOARD BY BOARD BASIS TO COMPENSATE FOR THE
"   MIXER BOX
"
"=====
"
A_0 = [XD30,XD10,XD28,XD04,XD60,XD35,XD15];
A_1 = [XD12,XD63,XD46,XD22,XD07,XD26,XD06];
A_2 = [XD65,XD45,XD64,XD49,XD25,XD08,XD59];
A_3 = [XD56,XD36,XD11,XD67,XD43,XD61,XD41];
A_4 = [XD38,XD00,XD29,XD14,XD70,XD52,SQR6];
A_5 = [XD20,XD09,XD47,XD32,XD17,XD34,SQR5];
A_6 = [XD02,XD18,XD03,XD50,XD44,XD16,SQR4];
A_7 = [XD55,XD27,XD21,XD68,XD53,XD69,SQR3];
A_8 = [XD37,XD54,XD39,XD24,XD62,XD51,SQR2];
A_9 = [XD19,XD01,XD57,XD42,XD71,XD33,SQR1];

LINK_VALUE = [1,0,1,0,1,1,0];
"=====
"
"   END OF MODIFICATION REGION
"
"=====

```

The VB code would extract the order of MCM data into the 10 (one per output) rows, sorted by time order. A unique link ID value would also be written. The new cut is similar, but not the same:

```

=====
"
"   THIS SECTION IS MODIFIED ON A BOARD BY BOARD BASIS TO COMPENSATE FOR THE
"   MIXER BOX
"
"=====
"
"LVDS half link Bit9 Bit8 Bit7 Bit6 Bit5 Bit4 Bit3 Bit2 Bit1 Bit0
Group_one   = [XD19, XD37, XD55, XD02, XD20, XD38, XD56, XD65, XD12, XD30]; "first tick bits
Group_two   = [XD01, XD54, XD27, XD18, XD09, XD00, XD36, XD45, XD63, XD10]; "second tick bits
Group_three = [XD57, XD39, XD21, XD03, XD47, XD29, XD11, XD64, XD46, XD28];
Group_four  = [XD42, XD24, XD68, XD50, XD32, XD14, XD67, XD49, XD22, XD04];
Group_five  = [XD71, XD62, XD53, XD44, XD17, XD70, XD43, XD25, XD07, XD60];
Group_six   = [XD33, XD51, XD69, XD16, XD34, XD52, XD61, XD08, XD26, XD35];
Group_seven = [SQR1, SQR2, SQR3, SQR4, SQR5, SQR6, XD41, XD59, XD06, XD15]; "last tick bits

FAKE_MAP0 = [0,0,0,0,0,0,0,0];
FAKE_MAP1 = [0,0,0,0,0,0,0,0];
FAKE_MAP2 = [0,0,0,0,0,0,0,0];
FAKE_MAP3 = [0,0,0,1,0,1,0,1];
FAKE_MAP4 = [0,0,0,0,0,0,0,SQR6];
FAKE_MAP5 = [0,0,0,0,0,0,0,SQR5];
FAKE_MAP6 = [0,1,0,0,0,0,0,SQR4];
FAKE_MAP7 = [0,0,0,0,0,0,0,SQR3];
FAKE_MAP8 = [0,0,0,0,0,0,0,SQR2];
FAKE_MAP9 = [0,0,0,0,0,0,0,SQR1];

LINK_VALUE = [1,0,1,0,1,1,0];

"=====
"
"   END OF MODIFICATION SECTION
"
"=====

```

The differences are:

- The MCM data is now sorted horizontally by bit and vertically by time, a transposition of the earlier method.
- A new set of data, sorted horizontally by time and vertically by bit, is included for the ‘fake track’ data. Note that the SQRn bits in the FAKE_MAP *should not be changed*.
- The Link ID is unchanged.

Program Control of the LVDS_MUX

The eight-bit control register of the LVDS_MUX is also changed to reflect increased functionality as shown in Table 1.

Bit	Name	Functionality
7	FORCE_STATE	Combines with LD_TEST signal (global selection of test mode vs. normal mode in LVDS Muxes) to select what data is presented
6	ENABLE_ID	If set, enables readback of the Link ID to the PIC.
5		No purpose on write, part of Link ID on read
4		No purpose on write, part of Link ID on read
3		No purpose on write, part of Link ID on read
2	ENBL_FAKE_TRACK	Controls assertion of Fake Track test pattern
1	FORCE_ODD	Controls DC force state of odd numbered (1,3,5,7,9) bits from LVDS_MUX
0	FORCE_EVEN	Controls DC force state of even numbered (0,2,4,6,8) bits from LVDS_MUX

Table 1

The bits act in concert with the global control LD_TEST (driven to all LVDS Muxes by the Clockgen in response to a PIC command) to select the mode of operation:

- FORCE_STATE == 0, LD_TEST == 0 sends actual MCM data.
- FORCE_STATE == 0, LD_TEST == 1 sends a walking-ones test pattern.
- FORCE_STATE == 1, LD_TEST == 0 sends a DC value controlled by the FORCE_ODD and FORCE_EVEN bits.
- FORCE_STATE == 1, LD_TEST == 1 sends the fake track pattern if ENBL_FAKE_TRACK is set, otherwise sends all zeroes.

Control of timing and synchronization data to Mixer and DFE boards

The Mixer and the DFE both rely on timing/synchronization data from the AFE in order to interpret the incoming data stream correctly. The 7.6MHz frame marker is always asserted regardless of the settings in the LVDS_MUX. The other synchronization bits are sent during real MCM data and fake track modes only. This allows the Mixer and the DFE to interpret the fake tracks identically to real ones.